TRANSLATOR'S NOTES ON FRENCH PATENT APPLICATION NO. 99 15892:

Please note the following sentence on page 13, lines 15-17: "Let Id be an object identifier in which the variables {W1, ...Wk} belong to the set {V1, ..., Vn} mentioned above." This represents a fair amount of guesswork by the translator, and should be confirmed with the client. The original French text is an incomplete sentence that translates literally as follows: "Let an object identifier Id in which the variables {W1, ...Wk} belonging to the set {V1, ..., Vn} mentioned above."

#9130139v1

In another embodiment, each indicator deployment agent is managed either by the agent machine that manages the configuration agent associated with the indicator deployment agent, or by a different agent machine.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention, along with its characteristics and advantages, will emerge more clearly through the reading of the description given in reference to the attached drawings, in which:

- Fig. 1 represents a simplified schema of a computer system in which the method of the invention can be applied.

- Fig. 2 represents the process for deploying a monitoring method.

DESCRIPTION OF THE INVENTIVE EMBODIMENTS

Setting up a monitoring of a computer system requires the configuration, then the deployment of this monitoring. A configuration of a monitoring is defined by a set of domains and indicators. A domain symbolically contains a set of equipment units of the computer system to be monitored. An indicator designates both the value characterizing the status or the operation of a set of equipment to be monitored, and the equation that makes it possible to calculate this value.

Configuring a monitoring consists of specifying for each indicator the domain or domains for which the indicator could be deployed. Deploying an indicator for a given domain consists of instantiating this indicator for each unit of equipment belonging to this domain. The method and the device according to the invention make it possible to perform this deployment.

The invention will now be described in reference to Figs.1 and 2.

As explained above, a computer system (1) comprises at least one local area network (10, 20) that communicates with a central system (2) or manager through a wide area network (3). Each local area network (10, 20) comprises at least one unit of computer equipment (101, 102, 201, 202) called a resource.

In the prior art, the monitoring of all the resources (101, 102, 201, 202) is handled by means of remote agents (SNMP or CMIP agents, depending on the management protocol in question) installed in each resource (101, 102, 201, 202). These agents allow a manager to collect information on the resources (101, 102, 201, 202) by measuring given parameters, for example by sending an SNMP request GetRequest to the SNMP agent in question, then to transmit the collected information to the manager (2) by means of a request, for example GetResponse in the SNMP world. The manager (2) then evaluates the indicators from this received information. It is understood that all the requests sent by the manager (2) in order to gather monitoring information and the responses to these requests from the agents produces

4

considerable congestion in the wide area network (3), consequently diminishing its performance.

According to the invention, all or part of the evaluation of the indicators is distributed at the local area network level (10, 20) to indicator agents.

5   To do this, for each indicator (In) characterizing a particular operation or status of a resource (101, 102, 201, 202), a specific indicator agent (A1, A2, B1, B2) is installed in the resource (101, 102, 201, 202). In the exemplary embodiment represented in Fig. 1, each agent (A1, A2, B1, B2) is responsible for evaluating a different indicator (In). In other words, each indicator (In) makes it possible to determine a different operation or a different status in each resource (101, 102, 201, 202). However, each agent (A1, A2, B1, B2) described in reference to Fig. 1 can be present, for example, in all of the resources (101, 102, 201, 202) or all the resources of the same type in the computer system (1).

For example, a first agent (A1) can evaluate an indicator (IA1) whose formula makes it possible to know, for example, the number of alarms per minute emitted by a first resource (101). It is understood that this agent (A1) can be installed in all the resources (101, 102, 201, 202) of a computer system (1).

The indicator agents (A1, A2, B1, B2) are elementary programming and execution entities. The indicator agents (A1, A2, B1, B2) are autonomous in order to give the structure the property of scalability by making it easy to add or delete one or several indicator agents in
20   the architecture. They are made autonomous by correctly defining the interface for these agents. They are also autonomous in terms of communication. They communicate using notifications and not procedure calls, which makes it possible to open up the architecture. This communication is asynchronous, thereby making it possible to obtain a better parallelization of the indicator agents and hence a better scalability of the architecture.

25   The indicator agents (A1, A2, B1, B2) can be persistent objects, and their status is derived from data that, in this case, can exist outside any execution structure and can be stored in persistent E^2PROM or hard disk memories. An indicator agent (A1, A2, B1, B2) is an independent machine that reacts to notifications.

Each agent (A1, A2, B1, B2) is an object of a class and is identified by a respective
30   identifier id(A1), id(A2), id(B1), id(B2) that is unique in the computer system (1) in order for communications between indicator agents to be possible, and that makes it possible to locate them. An agent can be created in a remote server. The creation entity is responsible for creating the identifier. The structure of the identifier must take into account the static location

5

of the agents in order to allow the system to forward the notifications, and to allow local "production" of the identifiers for the remote agents. The identifier of an agent comprises the following three parts:

- the identification (idsac) of the agent server hosting the creation agent,
- the identification (idsal) of the agent server hosting the agent created, and
- a postmark (el), local to the agent server hosting the creation agent.

A "factory" agent is present in all of the agent servers and is responsible for creating objects in the server It makes it possible to manage the creation of remote agents.

According to the invention, each indicator agent (A1, A2, B1, B2) is responsible for evaluating one indicator (In). In order to do this, it must comprise properties that make it possible to collect a measurement (Mn) on the resource (101, 102, 201, 202) with which it is associated, and communication properties in order to request or receive, as necessary, at least one other indicator evaluated by another indicator agent (A1, A2, B1, B2) according to the invention, for example by means of notifications. These properties are obtained by means of specific program modules constituting each indicator agent (A1, A2, B1, B2). Thus, an indicator agent (A1, A2, B1, B2) comprises a collection module and/or a communication module. Likewise, an indicator agent (A1, A2, B1, B2) also comprises an evaluation module for calculating the indicator itself from the formula, using the information collected by the other two modules. In this evaluation module, the equation that defines the value of an indicator (IA1 for example) can refer to a possibly empty set of other indicators (IA2, IB1, for example).

The indicator agents use, for example, a communication protocol that is preferably asynchronous, which means that when an indicator agent sends a request for collecting a measurement on a resource of the system or a notification to another indicator agent, it does not wait for the response to this request or notification before possibly sending a new request or notification. In other words, the sending of requests and notifications is performed in parallel. Likewise, the management protocol can be asynchronous. For example, the asynchronous management protocol used is the protocol SNMP or the protocol CMIP.

However, it is possible to use a synchronous protocol, but the performance in that case is inferior, particularly in terms of speed in evaluating the indicators. Moreover, it is appropriate to install an asynchronous device for collecting and storing measurements so that the indicator agent that sent the request need only consult the memory of this measurement

collecting device. This device is, for example, a particular management information base (MIB), called an Rmon MIB by one skilled in the art.

Each indicator agent manages a so-called subscriber list on which the names of other indicator agents according to the invention may be written. This list is stored in the storage means of the computer equipment unit associated with the indicator agent, for example in the form of a table (1010, 2010). An indicator agent A1 is written on this list by sending a specific so-called subscription notification "Subscribe (id(A1) Management Information (A1))" to another indicator agent B1, which calculates the indicator (IB1). This notification includes as parameters a piece of so-called management information that allows the sending agent to create an association between a propagation of a value modification and the other indicator agent (B1), and the identifier id(A1). Upon receiving a subscription notification, the destination agent (B1) processes the notification by writing into the subscriber table (2010) the identifier (IdA1) of the sending agent as well as the management information (Management Information (A1)) on the list. This list is consulted by the indicator agent (B1) that manages it, after the evaluation of the indicator (IB1) by the agent (B1). If the new value of the indicator is different from the value previously evaluated and stored by the agent, then the agent sends each agent written on the subscriber list (2010) and identified by the parameter (id(A1)) a value change notification (ValueChanged) comprising the new value (Val(IB1)) of the indicator (IB1). To do this, after the evaluation of the indicator (IB1), a comparison module of the indicator agent (B1) compares the new value of the indicator (Val(IB1)) to the value previously calculated (Valp(IB1)) and stored. If the two values are different, the comparison module activates a procedure for sending the value change notification (ValueChanged) to all of the agents, for example A, written on the subscriber list (2010) then records the new value of the indicator in the storage means of its computer equipment unit. The value change notification comprises as parameters the new value of the indicator (Val(IB1)) and the management information (Management Information (A1)) of the target agent, so that the target agent can assign the value received to the indicator awaited.

According to the invention, the monitored domain constituted by the set of resources (101, 102, 201, 202) is organized into a plurality of subdomains (d1={A1, A2}; d2={B1, B2}). Each of the subdomains contains either a set of resources to be monitored, and/or a set of subdomains.

A subdomain is also characterized by the indicator agents running in the resources constituting the subdomain. For each subdomain, a set of agents called "synthesis agents" is

7

installed in a particular resource so as to constitute a synthesis node. This resource can be a dedicated monitoring resource or a general-purpose resource chosen because of its technical characteristics to support the synthesis node. Moreover, this resource could also be a given resource of the subdomain or of another subdomain. In fact, one resource can support the monitoring of several subdomains, for example ranging from 0 to n, if this resource is capable of handling the load produced by the calculations performed by a plurality of sets of synthesis agents.

Basically, the structure of a synthesis agent is identical to that of an indicator agent; the difference in terms of behavior and utilization lies mainly in the formula representing the indicator to be evaluated.

During the configuration of the monitoring of the monitored domain according to the invention, a naming service is used to define and maintain the associations between the name of the subdomain and the identification of the indicator agent responsible for evaluating an indicator of the subdomain.

In other words, the naming service (SN) supplies, for each subdomain, all of the indicator agents present in this subdomain and associates, for example in a table (4) or in an association file, each indicator agent (A1) with the indicator (Ia1) it calculates and with the subdomain (d1) to which it belongs. This naming service SN is then used during the configuration of each synthesis agent. This naming service can be either centralized or distributed. When the naming service is centralized, as represented in Fig. 1, the information relative to the association between a subdomain and an indicator agent is centralized in a single place in the monitored domain represented, for example, by a resource of the monitored domain.

When the naming service is distributed, the naming service associated with each agent machine has the names of the agents it manages, as well as the subdomain to which each agent belongs.

The agent machine manages a certain number of properties, i.e. the atomic aspect of an agent's reaction to a notification and the delivery of the notifications in a causal sending order.

The agent machine handles the distribution of the notifications, the transmission of the notifications and the management of the overall atomicity. The agent machine comprises various entities such as an execution engine and a communication channel, this channel having two queues, a local queue and an external queue.

The agent machine is distributed. The agent machines communicate with each other; the agents do not see this distribution since they communicate through notifications and address their local agent machine. An agent's reaction is always local.

The centralized or distributed naming service SN is necessary for the configuration of the synthesis agents (As). In essence, as a general rule, a synthesis agent is designed to evaluate an indicator whose formula includes at least one set-oriented operator such as a sum, an average, or the determination of a minimum or a maximum in one or more indicators of the subdomain with which the synthesis agent is associated. However, a synthesis agent can also evaluate an indicator representing the general operation of the subdomain with which it is associated.

Thus, the configuration of a synthesis agent requires the latter to subscribe to all the indicator agents of the subdomain monitored by the synthesis agent that evaluate the indicator or indicators contained in the formula of the indicator of the synthesis agent. To do this, the synthesis agent (As) sends the naming service (SN), for each indicator required for the evaluation of its own indicator, a notification requesting an indicator lookup (indicatorLookupReq) that specifies the name of the subdomain (d1) that the synthesis agent (As) monitors, as well as the names of the indicators required for the evaluation of its own indicator. In response to this notification, the naming service performs a search in order to find the names of all the agents that perform the evaluation of the requested indicator in the subdomain specified in the notification. The naming service constructs and then sends a response notification (indicatorLookupResp) to the requesting synthesis agent containing the name of the domain (d1) and the name of the indicator agent or agents (A1, A2), if they exist. This response notification is processed by the synthesis agent, which extracts the names of the indicator agents and activates, for each indicator agent extracted, the subscription procedure described above.

In a variant of embodiment, when no indicator agent corresponding to the subdomain name/indicator name association exists, the naming service can include means for storing the name of the synthesis agent that sent the notification and the name of the indicator requested. Thus, as soon as a new indicator agent is installed in the subdomain, it is detected by the detection means of the naming service (SN) or is declared to the latter by the installer. The naming service includes means for performing an update of the subdomain name/indicator name associations and for verifying whether the name of the indicator corresponds to a stored indicator name. If so, means in the naming service construct, then send, the response

9

notification (indicatorLookupResp) to the agents whose names have been previously stored that have requested the name of the agent responsible for the new indicator detected.

In another variant, the naming service uses an agent creation tool, such as an agent machine, to construct the agent requested. In other words, the naming service constructs an

5   indicator agent that evaluates the requested indicator in the specified subdomain.

According to the invention, the number of resources per subdomain is lower than a predetermined maximum number. This number is determined as a function of the monitoring policy chosen. For example, the maximum number of resources per machine is determined so that the cost of calculating the indicators of the synthesis agents is as low as possible, in order

10   to reduce the calculation load in the resource or resources supporting the synthesis agents. Another possibility consists of determining the maximum number of resources per domain so that the number of synthesis nodes is as low as possible, in order to reduce the number of resources responsible for monitoring and to concentrate the information representing the monitoring.

15   According to Fig. 2, the deployment of a monitoring configuration consists of performing the instantiation, i.e. the creation, of the indicator agents for the indicators defined by the list {(d1.I1), ..., (di.Ij), ..., (dn.In)}, in which the indicator Ij must be evaluated in the subdomain di. To do this, the deployment method uses an agent called a configuration deployment agent (ADC). This configuration deployment agent handles the creation of agents

20   called configuration agents (AC). Thus, for each resource of a monitored subdomain, the configuration deployment agent (ADC) creates a configuration agent (ACa, ACb), which handles the process of creating the indicator agents (Ai, Bi, Ci) specified by the configuration for the resource that has been assigned to it. Thus, the deployment of the various configurations is parallelized in each of the subdomains to be monitored. In essence, the

25   configuration deployment agents are created for each subdomain to be monitored in parallel fashion.

The configuration deployment agents (ADC) and the configuration agents (AC) have the same characteristics as the indicator agents, i.e. they are also managed by at least one agent machine of at least one resource of the domain to be monitored. The location of the

30   configuration agents is of little importance; the various configuration agents can reside in the same resource or can be deployed in different resources, including even monitored resources.

As explained above, a configuration agent (AC) handles the creation of the indicator agents specified by the configuration for a given resource M belonging to at least one of the

10

subdomains (di) for which this indicator In must be created. To this end, for any indicator In of the configuration that can be instantiated in the resource M, the configuration agent creates an agent called an indicator deployment agent (ADI_A, ADI_B, Fig. 2) responsible for the deployment of the indicator or indicators I in the resource M.

5 The indicator deployment agent (ADI) can be created in the same resource as the configuration agent (AC) or in a different resource.

In a variant of embodiment in which the programming language used is the "Java" language, the configuration agents (AC) will induce the dynamic loading of the classes "I_Deployer" and "I_Indicator", using the mechanisms defined by Java runtime. When these classes are not present in the resource in which the configuration process is executed, the configuration agent receives an exception of the "class not found" type "ClassNotFoundException", which activates, at the configuration agent level, means for downloading the software elements it requires in the resource, thereby incrementally deploying the software elements required for the monitoring, from a minimal kernel. Thus, the monitoring method according to the invention provides a solution to the dual problem of configuring a distributed monitoring, i.e., deploying the software configuration and configuring the monitoring.

An indicator deployment agent (ADI) is an agent that determines, for a given type of indicator, the various combinations of the values of the variables for which the indicator will be instantiated. It therefore handles both the name resolution process (described below) and the creation of the indicator agents, as well as their declaration to the naming service (SN).

The names of the objects referenced by the indicator agent that calculates the indicators, as well as the identifications of the agents that calculate the indicators referenced during the calculation, are part of the parameters for the creation of any indicator agent during its instantiation by the indicator deployment agent.

For any indicator In, an indicator compiler, after analyzing the equation that defines the indicator, generates two object classes "I_Deployer" and "I_Indicator", which respectively correspond to the indicator deployment agents that deploy the instances of the class "I_Indicator" responsible for evaluating the indicator and to the indicator agents that evaluate the indicator. The class "I_Deployer" makes it possible to know which indicator agents identified by the class "I_Indicator" must be created, and makes it possible to declare to the naming service (SN) the indicator agents actually created.

An indicator deployment agent has explicit knowledge of the indicators and the identifiers of the objects referenced by the equation. Each of these object identifiers {Id1, ..., Idm} defines a structure wherein certain elements can be variables.

When the management protocol chosen is the asynchronous protocol SNMP, the indicator deployment agent executes the process (described below) for resolving the names of the objects referenced in the equation or the formula of the indicator and creates the corresponding indicator agents by determining the valid combinations of the values of the variables.

In essence, the equation that defines the calculation of the value of an indicator refers to objects identified by {Id1, ..., Idm} using a possibly empty set of variables {V1, ..., Vn}. Each object identifier Idi is associated with a set of variables {W1, ... Wk} belonging to the set {V1, ..., Vn}. Determining the first valid combination of the values of the variables Vi consists of applying a process (described below) for searching through the identifiers Idi, for example in the order 1 through m, in order to progressively instantiate all the variables {V1, ..., Vn} and hence to calculate the identifiers {Id1, ..., Idm}.

When all of the objects identified by {Id1, ..., Idm} in which the variables have been replaced with their corresponding values belonging to the combination of values {V1, ..., Vn} exist, the search process of the indicator deployment agent (ADI) verifies whether or not the constraint expressed in the values of the variables is satisfied. The indicator deployment agent (ADI) instantiates the indicator agent only when this last constraint is met. The objects actually referenced by the equation of the indicator are those identified by {Id1, ..., Idm} in which the variables have been replaced by their corresponding values belonging to the combination of values {V1, ..., Vn}. These objects are passed to the indicator agent as a parameters during the creation of the indicator agent.

In order to find the subsequent valid combinations of the variables Vi, the search process of the indicator deployment agent (ADI) searches for the index k for which there exists a subsequent element for Idk, for k varying between m and 1. If such a value of k does not exist, then the search process is terminated.

If such a value of k does exist, then the search process is applied to the identifiers Idi, in the order k+1 through m for example, in order to progressively instantiate all the variables {V1, ..., Vn} and hence to calculate the identifiers {Id1, ..., Idm}.

The search process therefore makes it possible to calculate all the objects {Id1, ..Idn} of the equation that represent an indicator.

12

The search process comprises a process for binding the variables, defined as follows, which makes it possible to determine the various variables of the identifiers Id_i. Let Id be an object identifier in which the variables $\{W1, \ldots Wk\}$ belong to the set $\{V1, \ldots, Vn\}$ mentioned above. The identifier Id has the form $a1.W1.a2.W2. \ldots .ak.Wk.ak+1$, in which the coefficients a_i have the structure of an object name in the ASN.1 sense of the term, which means that the coefficients a_i are constituted by a sequence of positive integers.

Let j be the index for which the variables with successive indices $\{W1, \ldots, Wj\}$ are already bound. Finding all of the objects that can be designated by this identifier consists of navigating through the instantiated MIB, searching for the objects by launching a request of the SNMP "GetNextRequest" type, using as a parameter an object identifier whose identifiers begin with the root $R = a1.W1.a2.W2. \ldots .aj.Wj$ and by applying an object identifier unifiability criterion, defined below. Let Id' be the response to the SNMP request GetNextRequest applied to the root R, if such an object exists. The object does not exist if the response to the request GetNextRequest indicates that the designated object does not exist, in which case the search is terminated.

If Id and Id' are unifiable according to the criterion defined below, then the variables $\{W1, \ldots, Wk\}$ are all bound, which means that they all have a value, and we have just found the identifier of an object that can be designated by Id, namely Id'.

In order to find the next valid combination of the variables $\{W1, \ldots, Wk\}$, knowing Id', it is necessary to reiterate the preceding process, using Id' as the starting point for the search, after having unbound, i.e., retrieved the previous values of, the variables W_i that were not bound at the very beginning of the preceding search process.

If Id and Id' are not unifiable and the identifier Id' begins with R, then the search continues as before, using Id' as the starting point and unbinding any variable W_i that was not bound at the very beginning of the preceding search process.

If Id' does not begin with R, then there is no longer any existing object whose identifier could be unified with Id. The search process is then terminated.

The unifiability criterion is defined as follows. Let Id' be a reference identifier (without a variable part) wherein all the elements are integer values; two identifiers Id and Id' are unifiable if both identifiers are the same size and if the identifiers $Id.a_i = Id'.a_i$ for any i belonging to $[1, k+1]$. Let $\{w1, \ldots, wk\}$ be the integer values of the identifier Id' that correspond to the variables $\{W1, \ldots, Wk\}$ of Id, i.e. that are located at the same position in

13

the corresponding identifier. The binding of the variables Wi takes place sequentially for indices varying between 1 and k through the following method:

if Wi is bound, i.e., if Wi already has a value, then this value must be equal to wi; if the latter condition is not satisfied, then Id and Id' are not unifiable and the binding of the variables {W1, …Wk} is not possible;

if Wi is free, i.e., if Wi does not yet have a value, then the variable Wi will be bound to the value wi.

For example, let Id = 1.2.3.W1.4.5.W2.6.7 and Id' = 1.2.3.10.4.5.20.6.7. These two object identifiers are unifiable for variables of W1 and W2 whose values are 10 and 20. On the other hand, if Id = 1.2.3.W1.4.5.W1.6.7 and Id' = 1.2.3.10.4.5.20.6.7, then Id and Id' are not unifiable because the variable W1 cannot have the values 10 and 20 at the same time.

The identifier unification process makes it possible both to verify that two identifiers are unifiable and to determine for which values of the variables that are still free prior to the start of the unification process this unification is possible.

When the name resolution process has exhausted all the valid combinations, the indicator deployment agent (ADI) becomes unnecessary if the monitored configuration is never changed.

When the software and/or hardware configuration of the monitored machine can change, for example, according to the invention, when the predetermined maximum number of machines per subdomain has been reached, the indicator deployment agents are reactivated in order to re-evaluate the hardware and software configuration by performing a search for the instantiable elements, instantiating only those of the indicator agents that don't already exist, and deleting those that no longer have any reason to exist.

In a variant of embodiment, when the programming language is Java, the creation of a configuration agent, an indicator deployment agent or an indicator agent consists of sending an agent creation request "agentCreateRequest" to the "factory" agent of the agent machine that manages the subdomain in which the agent must be deployed. This request includes as parameters the identification of the agent and the status of the agent. The status of the agent corresponds to the serialized Java object of this agent. When the "factory" agent receives this request, it deserializes the serialized Java object and thereby obtains the requested agent.

~~It should be clear to those skilled in the art that the present invention allows for embodiments in many other specific forms without going beyond the scope of application of the invention as claimed. Consequently, the present embodiments should be considered as~~

14

β

examples, but can be modified within the range defined by the scope of the attached claims, and the invention should not be limited to the details given above.